UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/750,601	12/30/2003	Leslie B. Lamport	224009	1522

41505 7590 11/25/2009

WOODCOCK WASHBURN LLP (MICROSOFT CORPORATION)
CIRA CENTRE, 12TH FLOOR
2929 ARCH STREET
PHILADELPHIA, PA 19104-2891

EXAMINER	
SCIACCA, SCOTT M	
ART UNIT	PAPER NUMBER
2446	

MAIL DATE	DELIVERY MODE
11/25/2009	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/750,601	LAMPORT, LESLIE B.	
	Examiner	Art Unit	
	Scott M. Sciacca	2446	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *09 September 2009*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-39* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-6,8-16 and 18-39* is/are rejected.

7)☒ Claim(s) *7 and 17* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *30 December 2003* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____.
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

This office action is responsive to communications filed on September 9, 2009. Claims 1, 9, 19 and 31 have been amended. Claims 1-39 are pending in the application.

Continued Examination Under 37 CFR 1.114

1. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on September 9, 2009 has been entered.

Claim Rejections - 35 USC § 103

2. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

3. Claims 1-4, 6, 8-14, 16, 18-24, 26-30, 31-37 and 39 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 5,261,085 (hereinafter, '085) in view of Paxos Made Simple (hereinafter, Paxos), from Applicant's IDS dated December 30, 2003 and US 6,532,494 (hereinafter, '494).

Regarding Claim 1, '085 teaches a method for selecting a value in a distributed computing system, the method comprising:

receiving at a first computing device from a first client a first message comprising a first proposed value and a first client identifier corresponding to the first client (*"One of the processes in the network is designated as a leader, and it sends ballots with proposed commands to the other processes"* – See Col. 3, lines 14-16; *"The preliminary protocol for conducting a single ballot initiated by process p is follows"* – See Col. 4, lines 56-57);

voting at the computing device for the first proposed value (*"In each ballot, a process has the choice of either voting for the proposed command or not voting"* – See Col. 3, lines 16-18);

transmitting from the first computing device a first indication of the voting for the first proposed value to one or more devices (*"$MaxVote(b,q,\beta)_{dec}$ is the vote with the largest ballot number less than b among all the votes cast by process q, or $null_q$ if process q did not vote in any ballot numbered less than b. The leader obtains $MaxVote(b,q,\beta)_{dec}$ from each process q by an exchange of messages"* – See Col. 4, lines 51-55); and

transmitting from the computing device a first result of the voting for the first proposed value to the first client (*"A process q responds to the receipt of the NextBallot(b) message by sending a LastVote(b,v) message to p"* – See Col. 4, lines 61-62).

'085 teaches clients having identifiers and determining if a second identifier is more dominant than a first identifier (*"One suitable method of choosing the leader is to select the process with the highest identification number"* – See Col. 8, lines 33-34). '085 does not explicitly teach that the voting for the first proposed value, the transmitting the first indication of the voting for the first proposed value, and the transmitting the first result are not performed if a second message is received at the computing device from a second client, the second message comprising a second proposed value and a second client identifier corresponding to a second client, the second identifier being more dominant than the first client identifier and the second proposed value having been previously voted for.

However, Paxos teaches a first client and a second client proposing values (*"Assume a collection of processes that can propose values"* – See p. 1). Paxos also discloses that a proposal numbered *n* is "accepted" by an acceptor if and only if the acceptor has not already accepted a request having a number greater (more dominant) than *n* (*"A proposer sends a proposed value to a set of acceptors. An acceptor may accept the proposed value. The value is chosen when a large enough set of acceptors have accepted it"* – See p. 2, lines 13-15; *"An acceptor can accept a proposal numbered n iff it has not responded to a prepare request having a number greater than n"* – See p. 5, lines 10-11). Paxos describes an acceptor accepting a proposed value in a fashion similar to the way '085 describes voting for a proposed value. It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the method of selecting a value disclosed by '085 to include not voting for a first

proposed value, not transmitting a first indication of the voting for the first proposed value, and not transmitting a first result if a second proposed value was proposed by a second client having a second client identifier that is more dominant than the first client identifier and the second proposed value was previously voted for. One of ordinary skill could easily apply the principle of comparing proposal numbers shown by Paxos to the teachings of '085 to end up with comparing identifiers of a first and second client, each of whom generates a proposal. Paxos describes a number of "safety requirements" for guaranteeing consensus in a distributed system on p. 1, lines 18-23. The step of comparing proposal numbers along with the respective outcomes of the comparison is necessary in order to satisfy the safety properties which guarantee consensus (See p. 5, lines 13-14).

'085 and Paxos do not explicitly teach that the second client identifier is determined to be more dominant than the first client identifier by evaluating the second identifier relative to the first identifier.

However, '494 teaches a voting scheme in a cluster computing system where a vote for a proposed value from a second client is accepted if the second client has a more dominant identifier relative to a first client (*"a distributed application, executing on the quorumless cluster 10, is afforded an opportunity to provide a vote from which it is determined which cluster partition will continue operating"* – See Col. 9, lines 66-67 & Col. 10, lines 1-2; *"At step 88, through an application program interface, (API), the cluster manager provides to the application a proposed figure of merit for the node the application is originating from"* – See Col. 10, lines 26-29; *"At step 90, the cluster*

managers for each of the nodes 12, 14, 16, 18 share the figure of merit data and determine, from the data, which cluster partition will continue operation" – See Col. 10, lines 44-46; *"In addition to the figures of merit provided from the cluster manager and application, tie-breaker algorithms may also be employed. These tie-breakers include which nodes joined the cluster first, which node has the highest or lowest node id etc"* – See Col. 10, lines 46-51).

It would have been obvious to modify '085-Paxos to accept a vote for a proposed value between a first client and a second client based on which client identifier has the more dominant client identifier relative to the other. When a first client and a second client propose different values in a vote, it effectively creates a tie. The scheme of selecting the node with the highest node ID in a tied situation is an effective way of breaking the tie (See '494, Col. 10, lines 46-51).

Regarding Claim 2, '085 further teaches that the first proposed value comprises a first function, and wherein the voting for the first proposed value comprises provisionally executing the first function in the first system step (*"To be issued, a command must be voted for by a majority of the processes in the system. Each issued command is stored by each of the processes in the majority set which voted for it"* – See Col. 2, lines 37-40).

Regarding Claim 3, '085 further teaches the voting for the first proposed value comprising changing a previous vote for the second proposed value if the second

proposed value was previously voted for and if the second client identifier is less dominant than the first client identifier (*"the receipt of a NextBallot(b) message by a process q in step 2 may set nextbal(q) to a value that prevents q from voting in step 4 for any previously initiated ballot"* – See Col. 7, lines 31-33).

Regarding Claim 4, '085 further teaches the first proposed value comprising a first function identified by a first function identifier (*"one of the processes send a message containing its identification number to every other process"* – See Col. 8, lines 36-37), and wherein the voting for the first proposed value comprises executing the first function in the first system step (*"Steps 1-4 thus contain the protocol for initiating a ballot and voting on it. In step 5, the results of the balloting are determined, and in step 6 the command is declared to be issued"* – See Col. 5, lines 59-62) unless the first function identifier is equivalent to a second function identifier that identifies a second function, wherein the second function was executed in a second system step that preceded the first system step (*"Receiving multiple copies of a message can cause an action to be repeated. Except in step 3, performing the action a second time has no effect. For example, sending several Voted(b,q) messages in step 4 has the same effect as sending just one. The repetition of step 3 is prevented by using the entry made in stable storage when it is executed. Thus, the consistency condition is maintained even if the same message is received several times"* – See Col. 5, lines 50-58).

Regarding Claim 6, '085 further teaches:

receiving a message (*"The network can be of any suitable type or configuration which permits messages to be sent between any two computers on the network"* – See Col. 3, lines 9-11), wherein the message is part of a fault tolerant consensus algorithm (*"This invention pertains generally to distributed computing systems and, more particularly, to a distributed system and method utilizing a state machine approach and having the ability to continue working despite the occurrence of a fault such as a processor stopping or running slowly"* – See Col. 1, lines 8-13);

ignoring additional proposed values from the first client (*"For example, a process q in the majority set can ignore a NextBallot(b) message"* – See Col. 5, lines 41-42); and

participating in the fault tolerant consensus algorithm (*"consistency is maintained despite the failure of any number of processes and communication paths"* – See Col. 10, lines 64-66).

Regarding Claim 8, '085 further teaches:

transmitting one or more polling messages to initiate a fault tolerant consensus algorithm (*"The preliminary protocol for conducting a single ballot initiated by process p is follows: …. 1. Process p (the leader) chooses a new ballot number b and sends a NextBallot(b) message to some set of processes"* – See Col. 4, lines 56-60);

receiving one or more vote indication messages in response to the one or more polling messages (*"2. A process q responds to the receipt of the NextBallot(b) message by sending a LastVote(b,v) message to p"* – See Col. 4, lines 61-62); and

selecting, as a third proposed value, any value if the one or more vote indication messages indicate that at least one device has not previously voted or if the one or more vote indication messages indicate two or more different possibly selected proposed values (*"any command for which one of the processes has previously voted but does not have a command number is broadcast as a proposed command in a new ballot"* – See Col. 2, lines 48-51), wherein a possibly selected proposed value was previously voted for by a device and was proposed by a client having a most dominant client identifier among all clients whose proposals were received by the device, and wherein further the third proposed value is proposed using the fault tolerant consensus algorithm (*"The selection requirement is then satisfied by having one of the processes send a message containing its identification number to every other process every T-11 msec, for some suitable choice of T"* – See Col. 8, lines 35-38).

Regarding Claim 9, '085 teaches a computer-readable storage medium having computer-executable instructions for selecting a value in a distributed computing system, the computer-executable instructions performing steps comprising:

receiving at a computing device from a first client a first message comprising a first proposed value and a first client identifier corresponding to the first client (*"One of the processes in the network is designated as a leader, and it sends ballots with proposed commands to the other processes"* – See Col. 3, lines 14-16; *"The preliminary protocol for conducting a single ballot initiated by process p is follows"* – See Col. 4, lines 56-57);

voting at the computing device for the first proposed value (*"In each ballot, a process has the choice of either voting for the proposed command or not voting"* – See Col. 3, lines 16-18);

transmitting from the first computing device a first indication of the voting for the first proposed value to one or more devices (*"$MaxVote(b,q,\beta)_{dec}$ is the vote with the largest ballot number less than b among all the votes cast by process q, or $null_q$ if process q did not vote in any ballot numbered less than b. The leader obtains $MaxVote(b,q,\beta)_{dec}$ from each process q by an exchange of messages"* – See Col. 4, lines 51-55); and

transmitting from the first computing device a first result of the voting for the first proposed value to the first client (*"A process q responds to the receipt of the NextBallot(b) message by sending a LastVote(b,v) message to p"* – See Col. 4, lines 61-62).

'085 teaches clients having identifiers and determining if a second identifier is more dominant than a first identifier (*"One suitable method of choosing the leader is to select the process with the highest identification number"* – See Col. 8, lines 33-34). '085 does not explicitly teach that the voting for the first proposed value, the transmitting the first indication of the voting for the first proposed value, and the transmitting the first result are not performed if a second message is received at the computing device from a second client, the second message comprising a second proposed value and a second client identifier corresponding to the second client, the second client identifier

being more dominant than the first client identifier and the second proposed value having been previously voted for.

However, Paxos teaches a first client and a second client proposing values (*"Assume a collection of processes that can propose values"* – See p. 1). Paxos also discloses that a proposal numbered *n* is "accepted" by an acceptor if and only if the acceptor has not already accepted a request having a number greater (more dominant) than *n* (*"A proposer sends a proposed value to a set of acceptors. An acceptor may accept the proposed value. The value is chosen when a large enough set of acceptors have accepted it"* – See p. 2, lines 13-15; *"An acceptor can accept a proposal numbered n iff it has not responded to a prepare request having a number greater than n"* – See p. 5, lines 10-11). Paxos describes an acceptor accepting a proposed value in a fashion similar to the way '085 describes voting for a proposed value. It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the steps of selecting a value disclosed by '085 to include not voting for a first proposed value, not transmitting a first indication of the voting for the first proposed value, and not transmitting a first result if a second proposed value was proposed by a second client having a second client identifier that is more dominant than the first client identifier and the second proposed value was previously voted for the same reasons as those given with respect to Claim 1.

'085 and Paxos do not explicitly teach that the second client identifier is determined to be more dominant than the first client identifier by evaluating the second identifier relative to the first identifier.

However, '494 teaches a voting scheme in a cluster computing system where a vote for a proposed value from a second client is accepted if the second client has a more dominant identifier relative to a first client (*"a distributed application, executing on the quorumless cluster 10, is afforded an opportunity to provide a vote from which it is determined which cluster partition will continue operating"* – See Col. 9, lines 66-67 & Col. 10, lines 1-2; *"At step 88, through an application program interface, (API), the cluster manager provides to the application a proposed figure of merit for the node the application is originating from"* – See Col. 10, lines 26-29; *"At step 90, the cluster managers for each of the nodes 12, 14, 16, 18 share the figure of merit data and determine, from the data, which cluster partition will continue operation"* – See Col. 10, lines 44-46; *"In addition to the figures of merit provided from the cluster manager and application, tie-breaker algorithms may also be employed. These tie-breakers include which nodes joined the cluster first, which node has the highest or lowest node id etc"* – See Col. 10, lines 46-51).

It would have been obvious to modify '085-Paxos to accept a vote for a proposed value between a first client and a second client based on which client identifier has the more dominant client identifier relative to the other for the same reasons as those given with respect to Claim 1.

Regarding Claim 10, '085 further teaches that the first proposed value comprises a first function, and wherein the voting for the first proposed value comprises provisionally executing the first function in the first system step (*"To be issued, a*

command must be voted for by a majority of the processes in the system. Each issued command is stored by each of the processes in the majority set which voted for it" – See Col. 2, lines 37-40).

Regarding Claim 11, '085 further teaches the voting for the first proposed value comprising changing a previous vote for the second proposed value if the second proposed value was previously voted for and if the second client identifier is less dominant than the first client identifier (*"the receipt of a NextBallot(b) message by a process q in step 2 may set nextbal(q) to a value that prevents q from voting in step 4 for any previously initiated ballot"* – See Col. 7, lines 31-33).

Regarding Claim 12, '085 further teaches the second proposed value comprising a second proposed function, and wherein the changing the previous vote comprises undoing a previous execution of the second proposed function (*"a process has the option not to vote, even if casting a vote is not precluded by a previous LastVote message"* – See Col. 5, lines 38-40).

Regarding Claim 13, '085 further teaches the second proposed value comprises a second proposed function, and wherein the changing the previous vote comprises allowing a previous provisional execution of the second proposed function to expire (*"the initiation of a new ballot can prevent any previously initiated ballot from succeeding"* – See Col. 7, lines 34-35).

Regarding Claim 14, '085 further teaches the first proposed value comprising a first function identified by a first function identifier (*"one of the processes send a message containing its identification number to every other process"* – See Col. 8, lines 36-37), and wherein the voting for the first proposed value comprises executing the first function in the first system step (*"Steps 1-4 thus contain the protocol for initiating a ballot and voting on it. In step 5, the results of the balloting are determined, and in step 6 the command is declared to be issued"* – See Col. 5, lines 59-62) unless the first function identifier is equivalent to a second function identifier that identifies a second function, wherein the second function was executed in a second system step that preceded the first system step (*"Receiving multiple copies of a message can cause an action to be repeated. Except in step 3, performing the action a second time has no effect. For example, sending several Voted(b,q) messages in step 4 has the same effect as sending just one. The repetition of step 3 is prevented by using the entry made in stable storage when it is executed. Thus, the consistency condition is maintained even if the same message is received several times"* – See Col. 5, lines 50-58).

Regarding Claim 16, '085 further teaches:

receiving at a computing device a message (*"The network can be of any suitable type or configuration which permits messages to be sent between any two computers on the network"* – See Col. 3, lines 9-11), wherein the message is part of a fault tolerant consensus method (*"This invention pertains generally to distributed computing systems*

and, more particularly, to a distributed system and method utilizing a state machine approach and having the ability to continue working despite the occurrence of a fault such as a processor stopping or running slowly" – See Col. 1, lines 8-13);

ignoring additional proposed values from the first client (*"For example, a process q in the majority set can ignore a NextBallot(b) message"* – See Col. 5, lines 41-42); and

participating in the fault tolerant consensus method (*"consistency is maintained despite the failure of any number of processes and communication paths"* – See Col. 10, lines 64-66).

Regarding Claim 18, '085 further teaches:

transmitting from the computing device one or more polling messages to initiate a fault tolerant consensus method (*"The preliminary protocol for conducting a single ballot initiated by process p is follows: …. 1. Process p (the leader) chooses a new ballot number b and sends a NextBallot(b) message to some set of processes"* – See Col. 4, lines 56-60);

receiving at the computing device one or more vote indication messages in response to the one or more polling messages (*"2. A process q responds to the receipt of the NextBallot(b) message by sending a LastVote(b,v) message to p"* – See Col. 4, lines 61-62); and

selecting, as a third proposed value, any value if the one or more vote indication messages indicate that at least one device has not previously voted or if the one or more vote indication messages indicate two or more different possibly selected

proposed values (*"any command for which one of the processes has previously voted but does not have a command number is broadcast as a proposed command in a new ballot"* – See Col. 2, lines 48-51), wherein a possibly selected proposed value was previously voted for by a device and was proposed by a client having a most dominant client identifier among all clients whose proposals were received by the device, and wherein further the third proposed value is proposed using the fault tolerant consensus method (*"The selection requirement is then satisfied by having one of the processes send a message containing its identification number to every other process every T-11 msec, for some suitable choice of T"* – See Col. 8, lines 35-38).

Regarding Claim 19, '085 teaches a computing device operating as part of a distributed computing system (Computer 12 – See Fig. 1), the computing device comprising a processing unit (*"Each of the computers includes at least a processor"* – See Col. 3, lines 3-4) and a network interface (*"a distributed system 11 in which the invention is employed has a set of computers 12 which are connected together by a network 13"* – See Col. 3, lines 1-3).

'085 teaches the network interface performing the steps comprising:

receiving at the computing device from the first client a first message comprising the first proposed value and a first client identifier corresponding to the first client (*"One of the processes in the network is designated as a leader, and it sends ballots with proposed commands to the other processes"* – See Col. 3, lines 14-16; *"The preliminary*

protocol for conducting a single ballot initiated by process p is follows" – See Col. 4, lines 56-57);

transmitting from the computing device a first indication of the voting for the first proposed value to one or more devices also operating as part of the distributed computing system (*"MaxVote(b,q,β)$_{dec}$ is the vote with the largest ballot number less than b among all the votes cast by process q, or null$_q$ if process q did not vote in any ballot numbered less than b. The leader obtains MaxVote(b,q,β)$_{dec}$ from each process q by an exchange of messages"* – See Col. 4, lines 51-55); and

transmitting a first result of the voting for the first proposed value to the first client (*"A process q responds to the receipt of the NextBallot(b) message by sending a LastVote(b,v) message to p"* – See Col. 4, lines 61-62).

'085 teaches clients having identifiers and comparing identifiers to determine if a second identifier is more dominant than a first identifier (*"One suitable method of choosing the leader is to select the process with the highest identification number"* – See Col. 8, lines 33-34). '085 does not explicitly teach that the comparing is performed if a second proposed value, proposed by a second client having the second client identifier, was previously voted for in a first system step. Similarly, '085 does not explicitly teach that the voting for the first proposed value, the transmitting the first indication of the voting for the first proposed value, and the transmitting the first result are not performed if a second message is received at the computing device from a second client, the second message comprising a second proposed value and a second client identifier corresponding to a second client, the second identifier being more

dominant than the first client identifier and the second proposed value having been previously voted for.

However, Paxos teaches a first client and a second client proposing values (*"Assume a collection of processes that can propose values"* – See p. 1). Paxos also teaches comparing identifiers for proposed values from a first and second client and voting for a first proposed value if the first client identifier is more dominant than the second client identifier (*"A proposer sends a proposed value to a set of acceptors. An acceptor may accept the proposed value. The value is chosen when a large enough set of acceptors have accepted it"* – See p. 2, lines 13-15; *"An acceptor can accept a proposal numbered n iff it has not responded to a prepare request having a number greater than n"* – See p. 5, lines 10-11).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the steps of selecting a value disclosed by '085 to include comparing a first client identifier to a second client identifier if a second proposed value, proposed by a second client having the second client identifier, was previously voted for in a first system step and voting for a first proposed value in the first system step if the first client identifier is more dominant than the second client identifier and the second proposed value was previously voted for the same reasons as those given with respect to Claim 1.

'085 and Paxos do not explicitly teach that the second client identifier is determined to be more dominant than the first client identifier by evaluating the second identifier relative to the first identifier.

However, '494 teaches a voting scheme in a cluster computing system where a vote for a proposed value from a second client is accepted if the second client has a more dominant identifier relative to a first client (*"a distributed application, executing on the quorumless cluster 10, is afforded an opportunity to provide a vote from which it is determined which cluster partition will continue operating"* – See Col. 9, lines 66-67 & Col. 10, lines 1-2; *"At step 88, through an application program interface, (API), the cluster manager provides to the application a proposed figure of merit for the node the application is originating from"* – See Col. 10, lines 26-29; *"At step 90, the cluster managers for each of the nodes 12, 14, 16, 18 share the figure of merit data and determine, from the data, which cluster partition will continue operation"* – See Col. 10, lines 44-46; *"In addition to the figures of merit provided from the cluster manager and application, tie-breaker algorithms may also be employed. These tie-breakers include which nodes joined the cluster first, which node has the highest or lowest node id etc"* – See Col. 10, lines 46-51).

It would have been obvious to modify '085-Paxos to accept a vote for a proposed value between a first client and a second client based on which client identifier has the more dominant client identifier relative to the other for the same reasons as those given with respect to Claim 1.

Regarding Claim 20, '085 further teaches the first proposed value comprising a first function, and wherein the voting for the first proposed value comprises provisionally executing the first function in the first system step (*"To be issued, a command must be*

voted for by a majority of the processes in the system. Each issued command is stored by each of the processes in the majority set which voted for it" – See Col. 2, lines 37-40).

Regarding Claim 21, '085 further teaches the voting for the first proposed value comprising changing a previous vote for the second proposed value if the second proposed value was previously voted for and if the second client identifier is less dominant than the first client identifier (*"the receipt of a NextBallot(b) message by a process q in step 2 may set nextbal(q) to a value that prevents q from voting in step 4 for any previously initiated ballot"* – See Col. 7, lines 31-33).

Regarding Claim 22, '085 teaches the second proposed value comprising a second proposed function, and wherein the changing the previous vote comprises undoing a previous execution of the second proposed function (*"a process has the option not to vote, even if casting a vote is not precluded by a previous LastVote message"* – See Col. 5, lines 38-40).

Regarding Claim 23, '085 further teaches the second proposed value comprising a second proposed function, and wherein the changing the previous vote comprises allowing a previous provisional execution of the second proposed function to expire (*"the initiation of a new ballot can prevent any previously initiated ballot from succeeding"* – See Col. 7, lines 34-35).

Regarding Claim 24, '085 teaches the first proposed value comprising a first function identified by a first function identifier (*"one of the processes send a message containing its identification number to every other process"* – See Col. 8, lines 36-37), and wherein the voting for the first proposed value comprises executing the first function in the first system step (*"Steps 1-4 thus contain the protocol for initiating a ballot and voting on it. In step 5, the results of the balloting are determined, and in step 6 the command is declared to be issued"* – See Col. 5, lines 59-62) unless the first function identifier is equivalent to a second function identifier that identifies a second function, wherein the second function was executed in a second system step that preceded the first system step (*"Receiving multiple copies of a message can cause an action to be repeated. Except in step 3, performing the action a second time has no effect. For example, sending several Voted(b,q) messages in step 4 has the same effect as sending just one. The repetition of step 3 is prevented by using the entry made in stable storage when it is executed. Thus, the consistency condition is maintained even if the same message is received several times"* – See Col. 5, lines 50-58).

Regarding Claim 26, '085 further teaches:

ignoring additional proposed values from the first client (*"For example, a process q in the majority set can ignore a NextBallot(b) message"* – See Col. 5, lines 41-42); and

participating in a fault tolerant consensus method (*"consistency is maintained despite the failure of any number of processes and communication paths"* – See Col. 10, lines 64-66); and

wherein the network interface performs further steps comprising:

receiving a message (*"The network can be of any suitable type or configuration which permits messages to be sent between any two computers on the network"* – See Col. 3, lines 9-11), wherein the message is part of the fault tolerant consensus method (*"This invention pertains generally to distributed computing systems and, more particularly, to a distributed system and method utilizing a state machine approach and having the ability to continue working despite the occurrence of a fault such as a processor stopping or running slowly"* – See Col. 1, lines 8-13).

Regarding Claim 27, '085 further teaches the participating in the fault tolerant consensus method comprising transmitting a possibly selected proposed value if a proposed value was previously voted for, wherein the possibly selected proposed value was previously voted for and was proposed by a client having a most dominant client identifier among all clients who proposed values to the computing device for a current system step (*"As part of this procedure, any command for which one of the processes has previously voted but does not have a command number is broadcast as a proposed command in a new ballot"* – See Col. 2, lines 48-51).

Regarding Claim 28, '085 further teaches:

selecting, as a third proposed value, any value if one or more vote indication messages indicate that at least one device has not previously voted or if the one or more vote indication messages indicate two or more different possibly selected proposed values (*"any command for which one of the processes has previously voted but does not have a command number is broadcast as a proposed command in a new ballot"* – See Col. 2, lines 48-51),

wherein a possibly selected proposed value was previously voted for by a device and was proposed by a client having a most dominant client identifier among all clients whose proposals were received by the device, and wherein further the third proposed value is proposed using the fault tolerant consensus method (*"The selection requirement is then satisfied by having one of the processes send a message containing its identification number to every other process every T-11 msec, for some suitable choice of T"* – See Col. 8, lines 35-38); and

wherein the network interface performs further steps comprising:

transmitting one or more polling messages to initiate the fault tolerant consensus algorithm (*"The preliminary protocol for conducting a single ballot initiated by process p is follows: …. 1. Process p (the leader) chooses a new ballot number b and sends a NextBallot(b) message to some set of processes"* – See Col. 4, lines 56-60); and

receiving the one or more vote indication messages in response to the one or more polling messages (*"2. A process q responds to the receipt of the NextBallot(b) message by sending a LastVote(b,v) message to p"* – See Col. 4, lines 61-62).

Regarding Claim 29, '085 further teaches the operating as part of the distributed computing system comprising operating as a client of the distributed computing system (*"a distributed system 11 in which the invention is employed has a set of computers 12 which are connected together by a network 13"* – See Col. 3, lines 1-3).

Regarding Claim 30, '085 further teaches the distributed computing system being comprised of devices that are also clients of the distributed computing system (*"a distributed system 11 in which the invention is employed has a set of computers 12 which are connected together by a network 13"* – See Col. 3, lines 1-3).

Regarding Claim 31, '085 teaches a conflict tolerant message delay reducing consensus method for use in computing environment comprising at least one dedicated client device and a distributed computing system implemented by one or more devices, wherein the computing system implements the conflict tolerant message delay reducing consensus method, the conflict tolerant message delay reducing consensus method comprising:

transmitting one or more proposed values from one or more clients, each of the one or more proposed values being transmitted in a message comprising one of the one or more proposed values and a client identifier corresponding to one of the one or more clients (*"One of the processes in the network is designated as a leader, and it sends ballots with proposed commands to the other processes"* – See Col. 3, lines 14-16);

voting, at one or more of the one or more devices implementing the distributed computing system, for a proposed value from among the one or more proposed values (*"In each ballot, a process has the choice of either voting for the proposed command or not voting"* – See Col. 3, lines 16-18), wherein the proposed value was proposed by a client having a most dominant client identifier from among the one or more clients proposing values (*"One suitable method of choosing the leader is to select the process with the highest identification number"* – See Col. 8, lines 33-34);

transmitting to one or more of the one or more devices implementing the distributed computing system an indication of the vote for the proposed value (*"MaxVote(b,q,β)$_{dec}$ is the vote with the largest ballot number less than b among all the votes cast by process q, or null$_q$ if process q did not vote in any ballot numbered less than b. The leader obtains MaxVote(b,q,β)$_{dec}$ from each process q by an exchange of messages"* – See Col. 4, lines 51-55); and

transmitting, to the client having the highest client identifier, a result of the vote for the proposed value (*"The preliminary protocol for conducting a single ballot initiated by process p is follows"* – See Col. 4, lines 56-57; *"A process q responds to the receipt of the NextBallot(b) message by sending a LastVote(b,v) message to p"* – See Col. 4, lines 61-62).

'085 teaches clients having identifiers and determining if a second identifier is more dominant than a first identifier (*"One suitable method of choosing the leader is to select the process with the highest identification number"* – See Col. 8, lines 33-34). '085 does not explicitly teach that the voting for the first proposed value, the transmitting

the first indication of the voting for the first proposed value, and the transmitting the first result are not performed if a second message is received at the computing device from a second client, the second message comprising a second proposed value and a second client identifier corresponding to a second client, the second identifier being more dominant than the first client identifier and the second proposed value having been previously voted for.

However, Paxos teaches a first client and a second client proposing values (*"Assume a collection of processes that can propose values"* – See p. 1). Paxos also discloses that a proposal numbered *n* is "accepted" by an acceptor if and only if the acceptor has not already accepted a request having a number greater (more dominant) than *n* (*"A proposer sends a proposed value to a set of acceptors. An acceptor may accept the proposed value. The value is chosen when a large enough set of acceptors have accepted it"* – See p. 2, lines 13-15; *"An acceptor can accept a proposal numbered n iff it has not responded to a prepare request having a number greater than n"* – See p. 5, lines 10-11). Paxos describes an acceptor accepting a proposed value in a fashion similar to the way '085 describes voting for a proposed value. It would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the method of selecting a value disclosed by '085 to include not voting for a first proposed value, not transmitting a first indication of the voting for the first proposed value, and not transmitting a first result if a second proposed value was proposed by a second client having a second client identifier that is more dominant than the first client identifier and the second proposed value was previously voted for. One of ordinary skill

could easily apply the principle of comparing proposal numbers shown by Paxos to the teachings of '085 to end up with comparing identifiers of a first and second client, each of whom generates a proposal. Paxos describes a number of "safety requirements" for guaranteeing consensus in a distributed system on p. 1, lines 18-23. The step of comparing proposal numbers along with the respective outcomes of the comparison is necessary in order to satisfy the safety properties which guarantee consensus (See p. 5, lines 13-14).

'085 and Paxos do not explicitly teach that the second client identifier is determined to be more dominant than the first client identifier by evaluating the second identifier relative to the first identifier.

However, '494 teaches a voting scheme in a cluster computing system where a vote for a proposed value from a second client is accepted if the second client has a more dominant identifier relative to a first client (*"a distributed application, executing on the quorumless cluster 10, is afforded an opportunity to provide a vote from which it is determined which cluster partition will continue operating"* – See Col. 9, lines 66-67 & Col. 10, lines 1-2; *"At step 88, through an application program interface, (API), the cluster manager provides to the application a proposed figure of merit for the node the application is originating from"* – See Col. 10, lines 26-29; *"At step 90, the cluster managers for each of the nodes 12, 14, 16, 18 share the figure of merit data and determine, from the data, which cluster partition will continue operation"* – See Col. 10, lines 44-46; *"In addition to the figures of merit provided from the cluster manager and application, tie-breaker algorithms may also be employed. These tie-breakers include*

which nodes joined the cluster first, which node has the highest or lowest node id etc" – See Col. 10, lines 46-51).

It would have been obvious to modify '085-Paxos to accept a vote for a proposed value between a first client and a second client based on which client identifier has the more dominant client identifier relative to the other for the same reasons as those given with respect to Claim 1.

Regarding Claim 32, '085 further teaches the one or more devices implementing the distributed computing system also acting as clients of the distributed computing system (*"a distributed system 11 in which the invention is employed has a set of computers 12 which are connected together by a network 13"* – See Col. 3, lines 1-3).

Regarding Claim 33, '085 further teaches the dedicated client device being identified by a least dominant client identifier (*"The selection requirement is then satisfied by having one of the processes send a message containing its identification number to every other process"* – See Col. 8, lines 35-37).

Regarding Claim 34, '085 further teaches determining that the distributed computing system has selected the proposed value when each of the one or more devices implementing the distributed computing system has voted for the proposed value (*"In order for a ballot to succeed and a command to be issued, a majority set of the processes in the system must vote for it"* – See Col. 3, lines 18-20).

Regarding Claim 35, '085 further teaches the proposed value comprising a function, and wherein the voting for the proposed value comprises provisionally executing the function in a system step (*"To be issued, a command must be voted for by a majority of the processes in the system. Each issued command is stored by each of the processes in the majority set which voted for it"* – See Col. 2, lines 37-40).

Regarding Claim 36, '085 teaches the voting for the proposed value comprising changing a previous vote if the previous vote was for a previously proposed value, proposed by a previous client having a client identifier that is less dominant than the client proposing the proposed value (*"the receipt of a NextBallot(b) message by a process q in step 2 may set nextbal(q) to a value that prevents q from voting in step 4 for any previously initiated ballot"* – See Col. 7, lines 31-33).

Regarding Claim 37, '085 teaches ending the conflict tolerant message delay reducing consensus algorithm and commencing a fault tolerant consensus algorithm if a failure is detected (*"The invention provides a system and method for implementing a distributed state machine in which consistency is maintained despite the failure of any number of processes and communication paths"* – See Col. 2, lines 22-25).

Regarding Claim 39, '085 further teaches identifying a possibly selected proposed value, wherein the possibly selected proposed value is any value if at least

one of the one or more devices implementing the distributed computing system did not previously vote (*"any command for which one of the processes has previously voted but does not have a command number is broadcast as a proposed command in a new ballot"* – See Col. 2, lines 48-51).

4. Claims 5, 15 and 25 are rejected under 35 U.S.C. 103(a) as being unpatentable over US 5,261,085 (hereinafter, '085) in view of Paxos Made Simple (hereinafter, Paxos) and US 6,532,494 (hereinafter, '494) and further in view of US 2003/0227392 (hereinafter, '392).

Regarding Claim 5, '085 in view of Paxos and '494 teaches the method of Claim 1. '085, Paxos and '494 do not explicitly teach the first proposed value comprising a first idempotent function, and wherein the voting for the first proposed value comprises executing the first idempotent function in the first system step even if the first idempotent function is equivalent to a second idempotent function that was executed in a second system step that preceded the first system step.

However, '392 discloses the use of idempotent operations (*"In general it is not easy to be certain which events were acted upon and which were lost. Furthermore, there is no inherent ordering of the processing of events through the real time semantics section 1630 and the core 1610. Replaying sufficient events to cover the largest possible loss works, provided all processing is idempotent and order independent"* – See [0263]) so that a first function idempotent function may be executed even though

an equivalent idempotent function has been executed previously (*"If this condition is met, the semantics of the operations invoked 2060 are such that if events are processed in different orders or if the same event is processed more than once, the ultimate system state will be identical"* – See [0263]).

It would have been obvious to one of ordinary skill at the time the invention was made to use idempotent functions with the distributed computing method taught by '085, Paxos and '494. '085 and Paxos generally disclose distributed computing systems where commands (functions) are distributed for execution by different machines on a network. '392 teaches that when idempotent functions are used, if the same function is processed (executed) more than once, then the end result will be identical (See [0263]).

Regarding Claim 15, '085 in view of Paxos and '494 teaches the computer-readable medium of Claim 9. '085, Paxos and '494 do not explicitly teaches the first proposed value comprising a first idempotent function, and wherein the voting for the first proposed value comprises executing the first idempotent function in the first system step even if the first idempotent function is equivalent to a second idempotent function that was executed in a second system step that preceded the first system step.

However, '392 discloses the use of idempotent operations (*"In general it is not easy to be certain which events were acted upon and which were lost. Furthermore, there is no inherent ordering of the processing of events through the real time semantics section 1630 and the core 1610. Replaying sufficient events to cover the largest possible loss works, provided all processing is idempotent and order independent"* –

See [0263]) so that a first function idempotent function may be executed even though an equivalent idempotent function has been executed previously (*"If this condition is met, the semantics of the operations invoked 2060 are such that if events are processed in different orders or if the same event is processed more than once, the ultimate system state will be identical"* – See [0263]).

It would have been obvious to one of ordinary skill at the time the invention was made to use idempotent functions with the distributed computing method taught by '085, Paxos and '494 for the same reasons as those given with respect to Claim 5.

Regarding Claim 25, '085 in view of Paxos and '494 teaches the computing device of Claim 19. '085, Paxos and '494 do not explicitly teaches the first proposed value comprising a first idempotent function, and wherein the voting for the first proposed value comprises executing the first idempotent function in the first system step even if the first idempotent function is equivalent to a second idempotent function that was executed in a second system step that preceded the first system step.

However, '392 discloses the use of idempotent operations (*"In general it is not easy to be certain which events were acted upon and which were lost. Furthermore, there is no inherent ordering of the processing of events through the real time semantics section 1630 and the core 1610. Replaying sufficient events to cover the largest possible loss works, provided all processing is idempotent and order independent"* – See [0263]) so that a first function idempotent function may be executed even though an equivalent idempotent function has been executed previously (*"If this condition is*

met, the semantics of the operations invoked 2060 are such that if events are processed in different orders or if the same event is processed more than once, the ultimate system state will be identical" – See [0263]).

It would have been obvious to one of ordinary skill at the time the invention was made to use idempotent functions with the distributed computing method taught by '085, Paxos and '494 for the same reasons as those given with respect to Claim 5.

5. Claim 38 is rejected under 35 U.S.C. 103(a) as being unpatentable over US 5,261,085 (hereinafter, '085) in view of Paxos Made Simple (hereinafter, Paxos) and US 6,532,494 (hereinafter, '494) and further in view of US 2002/0112198 (hereinafter, '198).

Regarding Claim 38, '085 does not explicitly teach the computing environment comprising a monitoring device and the failure being detected by a monitoring device. However, '198 does teach a monitoring device that detects a failure (*"At one extreme, the system administrator, operator or a third party (e.g., software for monitoring devices) detects a device failure"* – See [0058]). It would have been obvious to one of ordinary skill in the art at the time the invention was made to include a monitoring device within the computing environment taught by '085. Motivation for doing so would be to facilitate recovery in the event that a device fails (See paragraph [0058] of '198).

Response to Arguments

6. Applicant's arguments with respect to Claims 1, 9, 19 and 31 have been considered but are moot in view of the new grounds of rejection.

Allowable Subject Matter

7. Claims 7 and 17 both recite similar features and are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims. Similarly, independent Claims 19 and 31 would be allowable if the subject matter of either of Claims 7 or 17 and the base claims which they depend from is incorporated into Claims 19 and 31.

Conclusion

Any inquiry concerning this communication or earlier communications from the examiner should be directed to Scott M. Sciacca whose telephone number is (571) 270-1919. The examiner can normally be reached on Monday thru Friday, 7:30 A.M. - 5:00 P.M. EST.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Jeff Pwu can be reached on (571) 272-6798. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Scott M. Sciacca/
Examiner, Art Unit 2446

/Jeffrey Pwu/
Supervisory Patent Examiner, Art Unit 2446